MEDICAL INVENTORY OPTIMIZATION

PREPROCESSING AND EDA USING PYTHON

** PREPROCESSING

- Importing necessary libraries

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

- Reading the data using pandas read_csv function.

```python
data = pd.read_csv(r"C:\Users\yasar\OneDrive\Desktop\360 digitmg\Dataset\dataset_Copy.csv")
```

Checking top 5 rows of the data set

```python
data.head()
```

	Typeofsales	Patient_ID	Specialisation	Dept	Dateofbill	Quantity	ReturnQuantity	Final_Cost	Final_Sales	RtnMF
0	Sale	12018098765	Specialisation6	Department1	6-1-2022	1	0	55.406	59.260	0
1	Sale	12018103897	Specialisation7	Department1	7/23/2022	1	0	768.638	950.800	0
2	Sale	12018101123	Specialisation2	Department3	6/23/2022	1	0	774.266	4004.214	0
3	Sale	12018079281	Specialisation40	Department1	3/17/2022	2	0	40.798	81.044	0
4	Sale	12018117928	Specialisation5	Department1	12/21/2022	1	0	40.434	40.504	0

The data has 14218 rows and 14 columns.

- Creating a new data frame named cleaned and changed the Dateofbill data type to datetime.

```
cleaned = data.copy()
```

```
## Changing text to datetime data type
cleaned['Dateofbill'] = pd.to_datetime(cleaned['Dateofbill'])
```

- Checking for null values in the dataset.

```
## Checking for any null values
cleaned.isna().sum()
```

```
Typeofsales          0
Patient_ID           0
Specialisation       0
Dept                 0
Dateofbill           0
Quantity             0
ReturnQuantity       0
Final_Cost           0
Final_Sales          0
RtnMRP               0
Formulation        653
DrugName          1668
SubCat            1668
SubCat1           1692
dtype: int64
```

Formulation, DrugName, SubCat, SubCat1 columns has null values so they are replaced with string ‘unknown

```
## Replacing null with 'unknown'
cleaned.fillna('Unknown', inplace = True)
```

```
cleaned.isna().sum()
```

```
Typeofsales        0
Patient_ID         0
Specialisation     0
Dept               0
Dateofbill         0
Quantity           0
ReturnQuantity     0
Final_Cost         0
Final_Sales        0
RtnMRP             0
Formulation        0
DrugName           0
SubCat             0
SubCat1            0
```

’.

- Checking for any duplicates in the dataset if there are any we remove them

```python
## Checking for any duplicates in the dataset
total_duplicates = cleaned.duplicated().sum()
total_duplicates
```

26

So there are 26 duplicates in our data that we can remove them using drop_duplicates() function

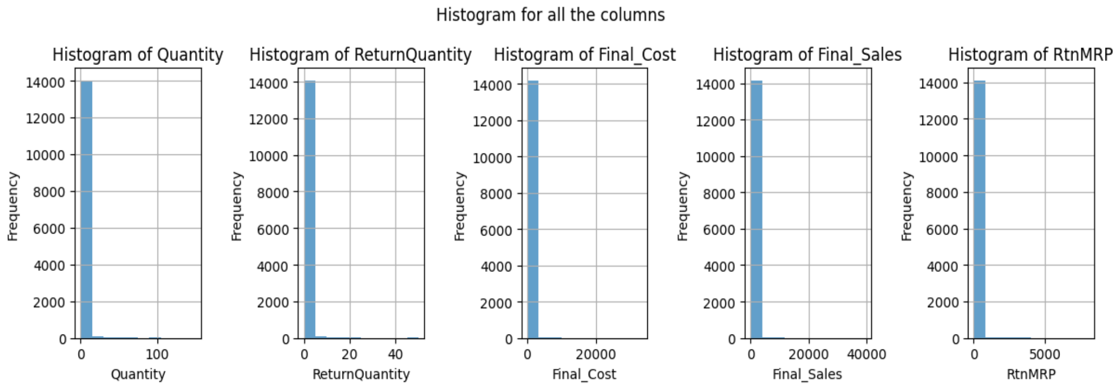
```python
## Removing all the duplicates
cleaned.drop_duplicates(inplace = True)
```

After removing duplicates data set has 14192 rows.

- Checking how the data is distributed and finding is there any outliers in the dataset

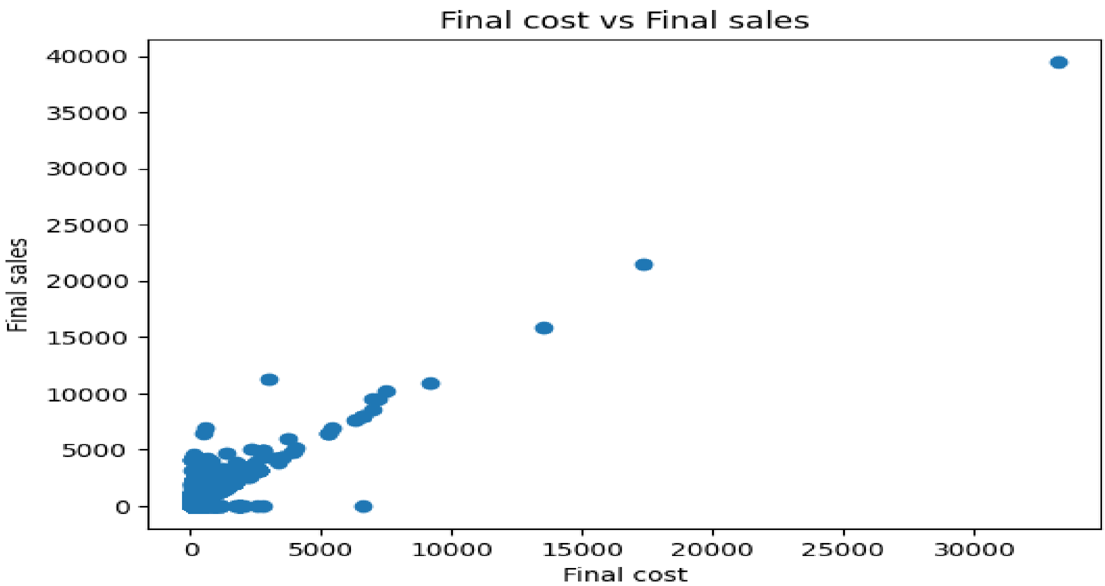
```python
columns = ['Quantity', 'ReturnQuantity', 'Final_Cost', 'Final_Sales', 'RtnMRP']
def histogram(cleaned):
    n_cols = 5
    n_rows = (len(columns) + n_cols - 1) // n_cols

    fig, axes = plt.subplots(nrows = n_rows, ncols = n_cols, figsize = (12, 4))
    fig.suptitle('Histogram for all the columns')

    axes = axes.flatten()

    for i, col in enumerate(columns):
        cleaned[col].hist(ax = axes[i], bins = 10, alpha = 0.7)
        axes[i].set_title(f'Histogram of {col}')
        axes[i].set_xlabel(col)
        axes[i].set_ylabel('Frequency')

    for ax in axes[len(columns):]:
        ax.set_visible(False)

    plt.tight_layout()

    plt.show()
```

```python
histogram(cleaned)
```



All the numerical columns have peak at a single place that denotes there is high frequency at that point but it doesn't mean that other values have 0 frequency. The lower the frequency of the point it is closer to the outlier.

The relationship between the final cost and final sales.



- We use normal distribution rule to eliminate outliers. Normal distribution has a rule that at 3 standard deviations there is 99.7% of the data. So we use 3 standard deviations to eliminate outliers. The formula to select lower boundary is mean – 3 * Standard deviation, For upper boundary mean + 3 * standard deviation.

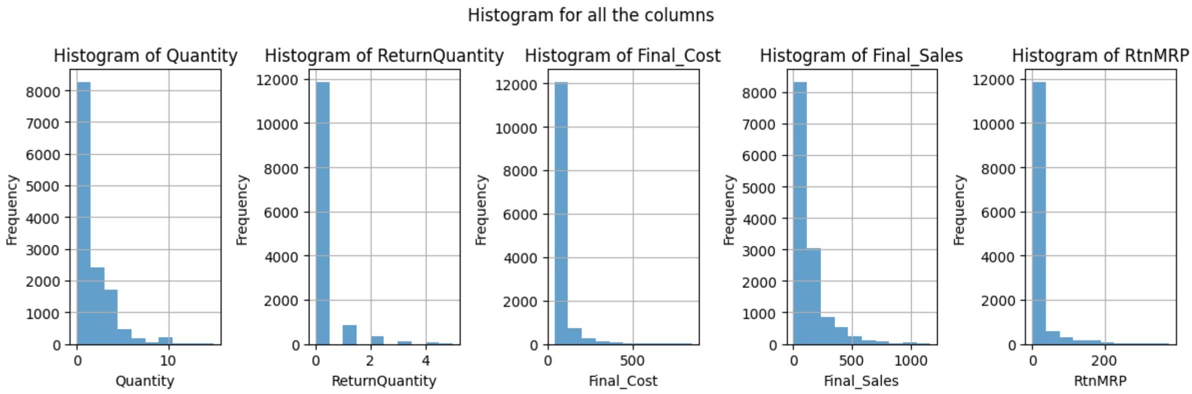
```
## Removing outliers

def outlier_remover(df):
    for i in columns:
        x_bar = df[i].mean()
        sigma = df[i].std()
        df = df[(df[i]>(x_bar-3*sigma)) & (df[i]<(x_bar+3*sigma))]
    return df
```

```
## creating a new data frame without outliers
cleaned_outliers = outlier_remover(cleaned)
```

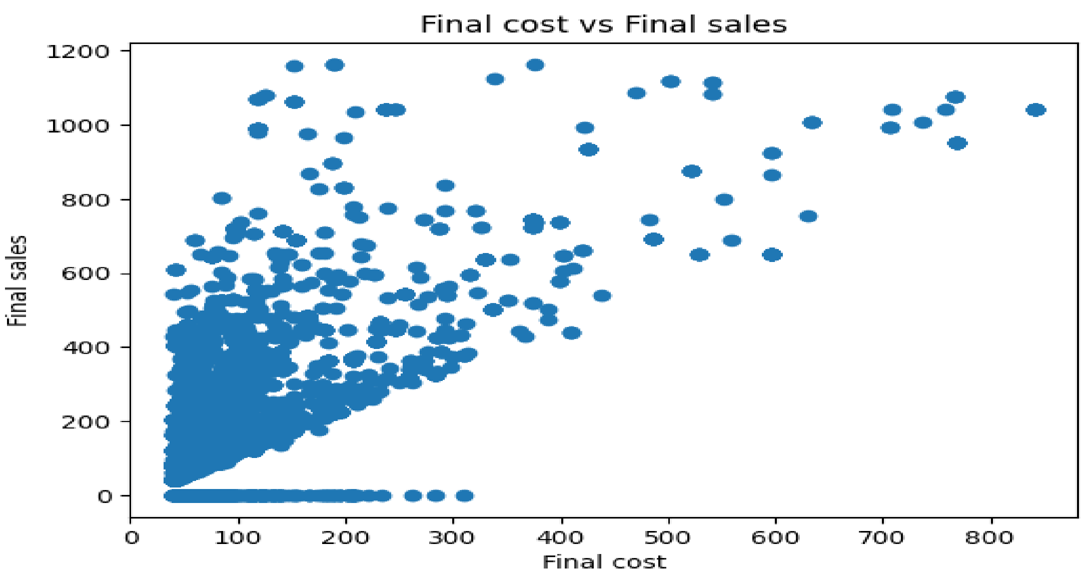
```
cleaned_outliers.shape
```

```
(13291, 14)
```

After removing outliers the data has 13291 records and the histogram looks like this



The relationship between final cost and final sales after outlier removal



- Creating a column for month that are extracted from the dateofbill column.

```
cleaned_outliers['month'] = cleaned_outliers['Dateofbill'].dt.month_name()
```

** EDA (Exploratory Data Analysis)

STATISTICAL INSIGHTS

- Here I performed descriptive statistics which covers measures of central tendancy and measures of dispersion.

```
columns = ['Quantity', 'ReturnQuantity', 'Final_Cost', 'Final_Sales', 'RtnMRP']
def desc_stats(data):
    for i in columns:
        print(i)
        print('-----------------------------------------------------------------')
        print("Measures of central tendancy")
        print()
        mean = data[i].mean()
        median = data[i].median()
        mode = data[i].value_counts().index.tolist()[0]
        print('Mean : ', mean)
        print('Median : ', median)
        print('Mode : ', mode)
        print()

        print("Measures of dispersion")
        print()
        standard_deviation = data[i].std()
        variance = data[i].var()
        Range = data[i].max() - data[i].min()
        print('Standard deviation : ', standard_deviation)
        print('Variance : ', variance)
        print('Range : ', Range)

        skewness = (3 * (mean - median))/standard_deviation
        kurtosis = data[i].kurtosis()
        print('Skewness : ', skewness)
        print('Kurtosis : ', kurtosis)
        print()
```

```
desc_stats(cleaned_outliers)
```

Here we get all the statistical values for each numerical column.

```
Quantity
------------------------------------------------------------------
Measures of central tendancy

Mean :  1.7793995937100293
Median :  1.0
Mode :  1

Measures of dispersion

Standard deviation :  1.7955550755271514
Variance :  3.2240180292513148
Range :  15
Skewness :  1.302215015846074
Kurtosis :  11.710676037581983

ReturnQuantity
------------------------------------------------------------------
Measures of central tendancy

Mean :  0.18185238131066134
Median :  0.0
Mode :  0

Measures of dispersion

Standard deviation :  0.613459689913864
Variance :  0.3763327911492142
Range :  5
Skewness :  0.8893121306936821
Kurtosis :  21.393556460881502
```

```
Final_Cost
-------------------------------------------
Measures of central tendancy

Mean :  72.57315311112782
Median :  52.32
Mode :  49.352

Measures of dispersion

Standard deviation :  64.68840569446536
Variance :  4184.589831291739
Range :  801.28
Skewness :  0.9392635153254667
Kurtosis :  43.94553554769619

Final_Sales
-------------------------------------------
Measures of central tendancy

Mean :  133.58799051990067
Median :  84.6
Mode :  0.0

Measures of dispersion

Standard deviation :  152.27823931389418
Variance :  23188.662168539624
Range :  1163.0
Skewness :  0.965101594435711
Kurtosis :  10.365260340785893

RtnMRP
-------------------------------------------
Measures of central tendancy

Mean :  12.488296140245279
Median :  0.0
Mode :  0.0

Measures of dispersion

Standard deviation :  43.587744037295415
Variance :  1899.8914302607818
Range :  378.338
Skewness :  0.859528045055036
Kurtosis :  22.542185231810976
```

- From the above statistical values there is a higher variance in the data except Quantity and Return Quantity columns and the skewness is closer to 1 that indicates that the data is skewed to the right and the kurtosis value is higher and it larger than the value 3 that shows the data has a high peak at a certain interval which is also called as leptokurtic.

Business insights

- What is present Bounce rate?

```
total_customers = cleaned_outliers['Patient_ID'].unique()
bounced_customers = cleaned_outliers[cleaned_outliers['Typeofsales'] == 'Return']['Patient_ID'].unique()
```

```
## calculating bounce rate
bounce_rate = (len(bounced_customers)/len(total_customers)) * 100
print('Bounce Rate : ', bounce_rate)
```

```
Bounce Rate :  23.352673021135516
```

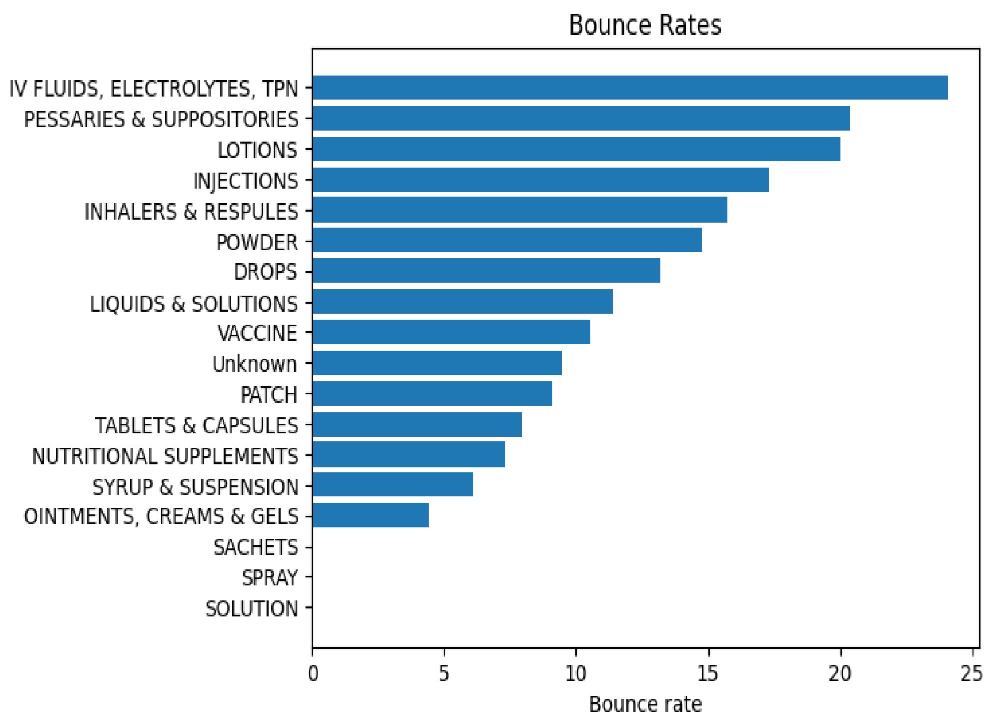
Bounce rate is 23% it is approximately a quarter portion of customers are returning back the medicines. This causes dissatisfaction to the customer.

- What are the categories that have higher bounce rates?

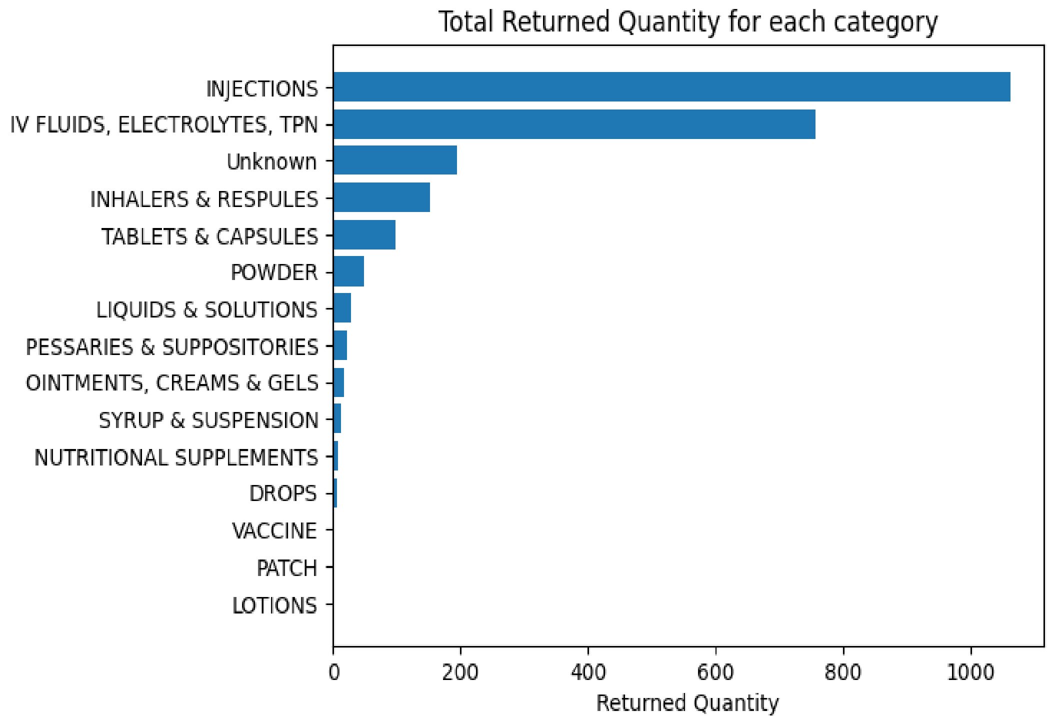
```
category = cleaned_outliers['SubCat'].unique() ## list of unique subcategories
cat_customers = []  ## Total customers for the category
bounced_cat = [] ## Total number of customers who returnes the product

for i in category:
    cat_customers.append(len(cleaned_outliers[cleaned_outliers['SubCat'] == i]
                         ['Patient_ID'].unique()))
    bounced_cat.append(len(cleaned_outliers[(cleaned_outliers['SubCat'] == i) &
                                            (cleaned_outliers['Typeofsales'] == 'Return')]
                       ['Patient_ID'].unique()))

## Creating a dataframe with subcat, totalcustomer, bouncedcustomers and bouncerate
category_bounce_rate = pd.DataFrame({'SubCat':category, 'Total_customers':cat_customers,
                                     'Bounced_customers':bounced_cat})

## Calculating bounce rate
category_bounce_rate['Bounce_rate'] = (category_bounce_rate['Bounced_customers']/
                                       category_bounce_rate['Total_customers'])*100

## Sorting th bounce rate in ascending order
sorted_cat = category_bounce_rate.sort_values(by = 'Bounce_rate', ascending = True)
```

	SubCat	Total_customers	Bounced_customers	Bounce_rate
17	SOLUTION	3	0	0.000000
13	SPRAY	12	0	0.000000
16	SACHETS	1	0	0.000000
6	OINTMENTS, CREAMS & GELS	341	15	4.398827
0	SYRUP & SUSPENSION	229	14	6.113537
4	NUTRITIONAL SUPPLEMENTS	109	8	7.339450
2	TABLETS & CAPSULES	1131	90	7.957560
15	PATCH	11	1	9.090909
5	Unknown	1139	108	9.482002
14	VACCINE	19	2	10.526316
9	LIQUIDS & SOLUTIONS	210	24	11.428571
10	DROPS	53	7	13.207547
11	POWDER	203	30	14.778325
8	INHALERS & RESPULES	369	58	15.718157
1	INJECTIONS	3191	552	17.298652
12	LOTIONS	5	1	20.000000
7	PESSARIES & SUPPOSITORIES	54	11	20.370370
3	IV FLUIDS, ELECTROLYTES, TPN	1699	409	24.072984

Creating a bar plot with categories and bounce rates

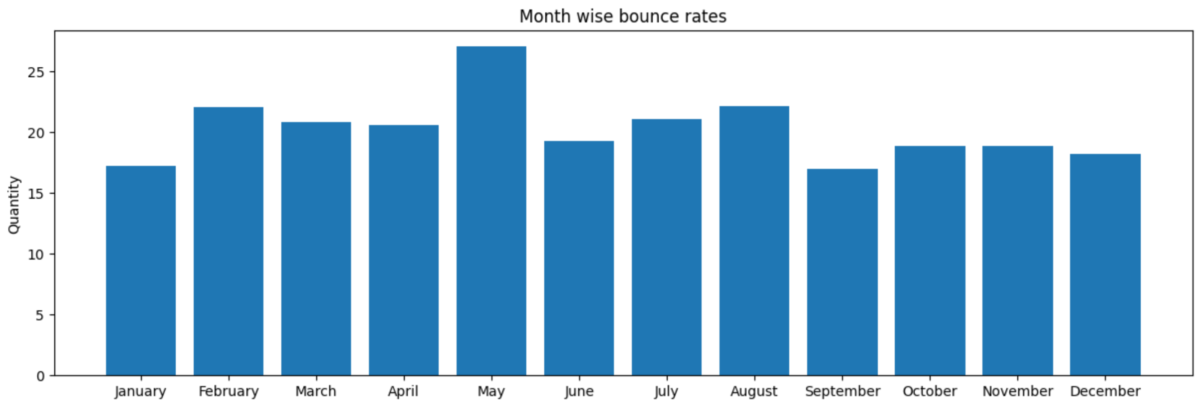


IV Fluids, Electrolytes,Tpn, Pessaries & Suppositories, Lotions are having bounce rate greater than 20%.



Injections category has more than 1000 returns which is a significant number that needed to be focused on the category.

- Are there any seasonal trends in bounce rates?



The above graph shows the monthly bounce rates we can see there seems no pattern in the bounce rates but there is a high in may month which crosses 25% bounce rate and January and September month has lesser bounce rate which is approximately 17%.

This shows the quantity returned for each category in each month

Conclusion :

- There is high bounce rates for products IV Fluids, Electrolytes, Tpn, Pessaries & Suppositories, Lotions and injections. Which needs some special attention and need to study deeper about those products and customer behaviour on these products.

- Based on the months the bounce rate is maximum i.e., 27% in the month of may and least i.e., 17% on the month of January and September. It may due to purchase of these type of products may be high on may month.

- Based on observation maximum quantity return is for injections category and maximum people returning the product is for IV Fluids, Electrolytes, Tpn category.

- Highest bounce rates for products injections and IV Fluids, Electrolytes, Tpn which are the part of formulation 1. The formulation 1 needs attention which may reveal some more findings and eventually reduces the bounce rate.